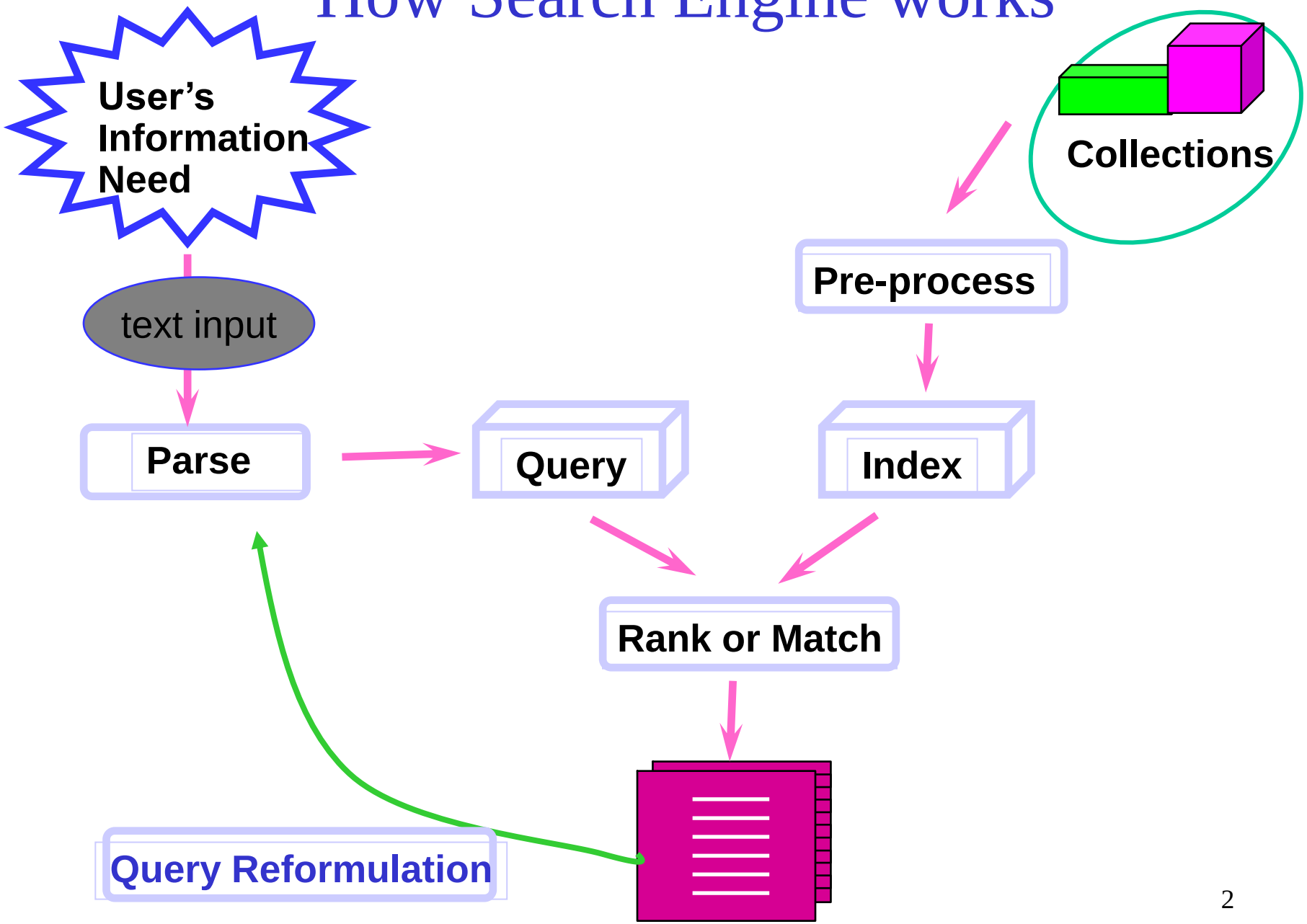
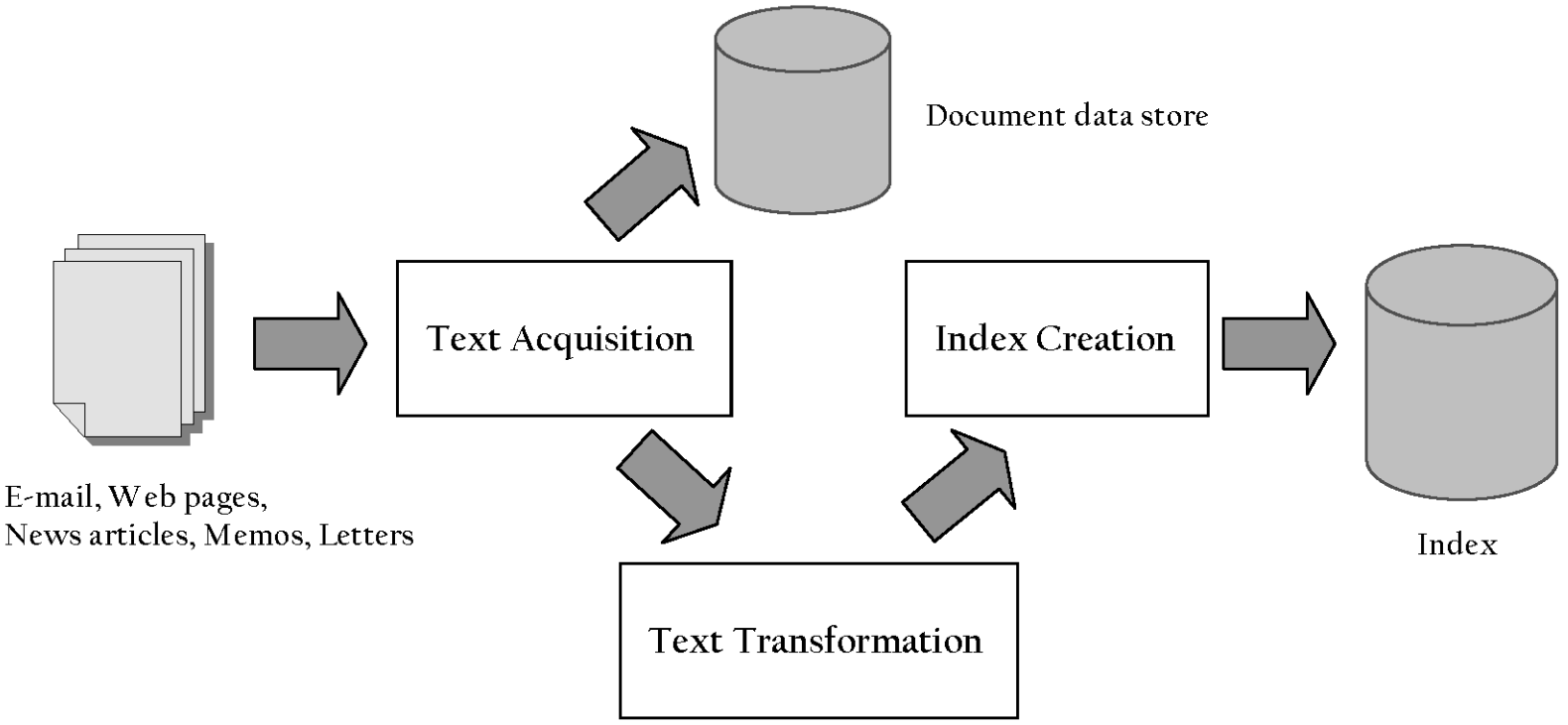
Web Engineering

Lecture 2
Search Engine Architecture
(1. Crawler)

How Search Engine works



Indexing Process



Indexing Process

- Text acquisition
  - identifies and stores documents for indexing
- Text transformation
  - transforms documents into *index terms* or *features*
- Index creation
  - takes index terms and creates data structures (*indexes*) to support fast searching

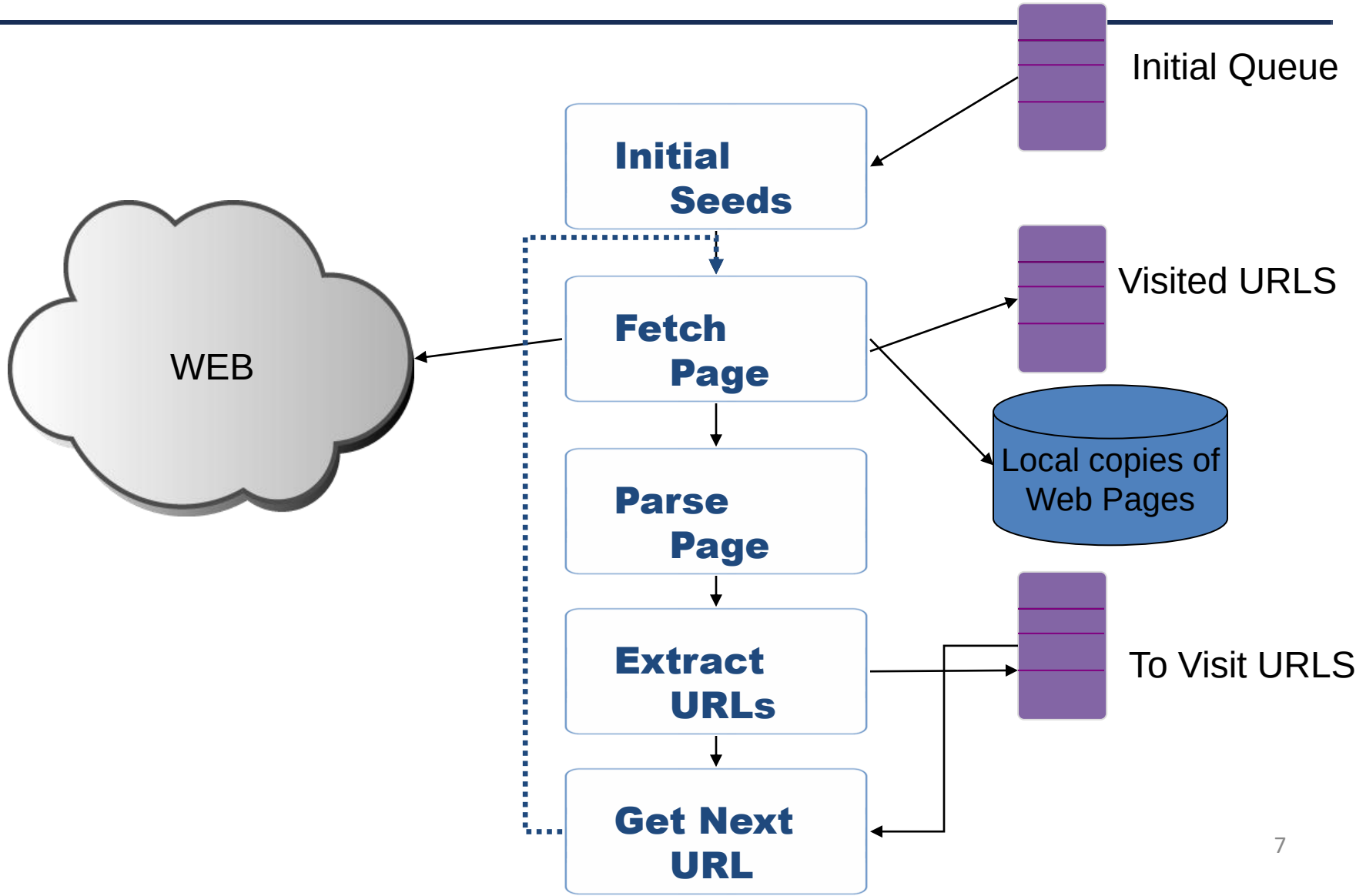
Text Acquisition: **Web Crawling**

- A **Web Crawler is a software for downloading pages** from the Web
- Also known as Web Spider, Web Robot, or simply Bot
- Web crawlers are mainly used to ***create a copy of all the visited pages for later processing*** by a search engine that will index the downloaded pages to provide fast searches.

How a Web crawler works?

- It starts with a ***list of URLs*** to visit, called the ***seeds***.
- As the crawler visits these URLs, it identifies all the ***hyperlinks*** in the page and adds them to the list of URLs to visit, called the ***crawl frontier***.
- URLs from the frontier are ***recursively visited*** according to a *set of policies*.

Processing Steps in Crawling

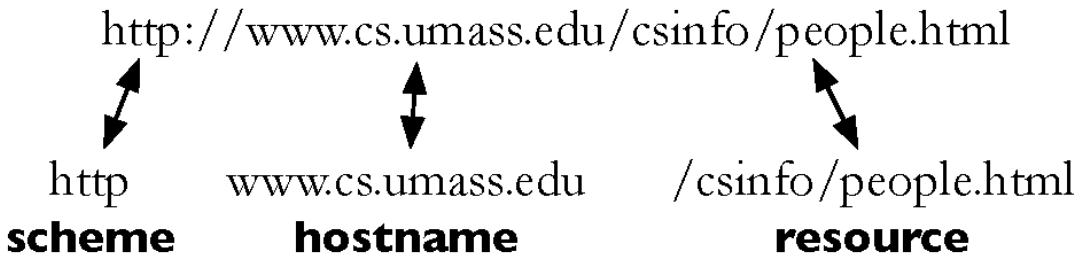


Processing Steps in Crawling

- Put a set of known sites on a queue (Seeds).
- Fetch the document at the URL.
- Parse the URL – HTML parser
  - Extract links from it to other docs (URLs)
- For each extracted URL
  - Ensure it passes certain URL filter tests
  - Check if it already exists in the frontier (duplicate URL elimination)

Retrieving Web Pages

- Every page has a unique *uniform resource locator* (URL)
- Web pages are stored on web servers that use HTTP to exchange information with client software
- e.g.,



Parsing: URL normalization

- When a fetched document is parsed, some of the extracted links are *relative* URLs
- E.g., http://en.wikipedia.org/wiki/Main_Page has a relative link to /wiki/Wikipedia:General_disclaimer which is the same as the absolute URL http://en.wikipedia.org/wiki/Wikipedia:General_disclaimer
- During parsing, must normalize (expand) such relative URLs

Filters & Robots.txt

- Protocol for giving spiders ("robots") limited access to a website:
  - www.robotstxt.org/wc/norobots.html
- Website announces its request on what can(not) be crawled
  - For a server, create a file `/robots.txt`
  - This file specifies access restrictions

Filters & Robots.txt

- Even crawling a site slowly will anger some web server administrators, who object to any copying of their data
- Robots.txt file can be used to control crawlers

```
User-agent: *
Disallow: /private/
Disallow: /confidential/
Disallow: /other/
Allow: /other/public/

User-agent: FavoredCrawler
Disallow:

Sitemap: http://mysite.com/sitemap.xml.gz
```

How does a crawler visit web pages?

Crawl frontiers are visited in one of two strategies:

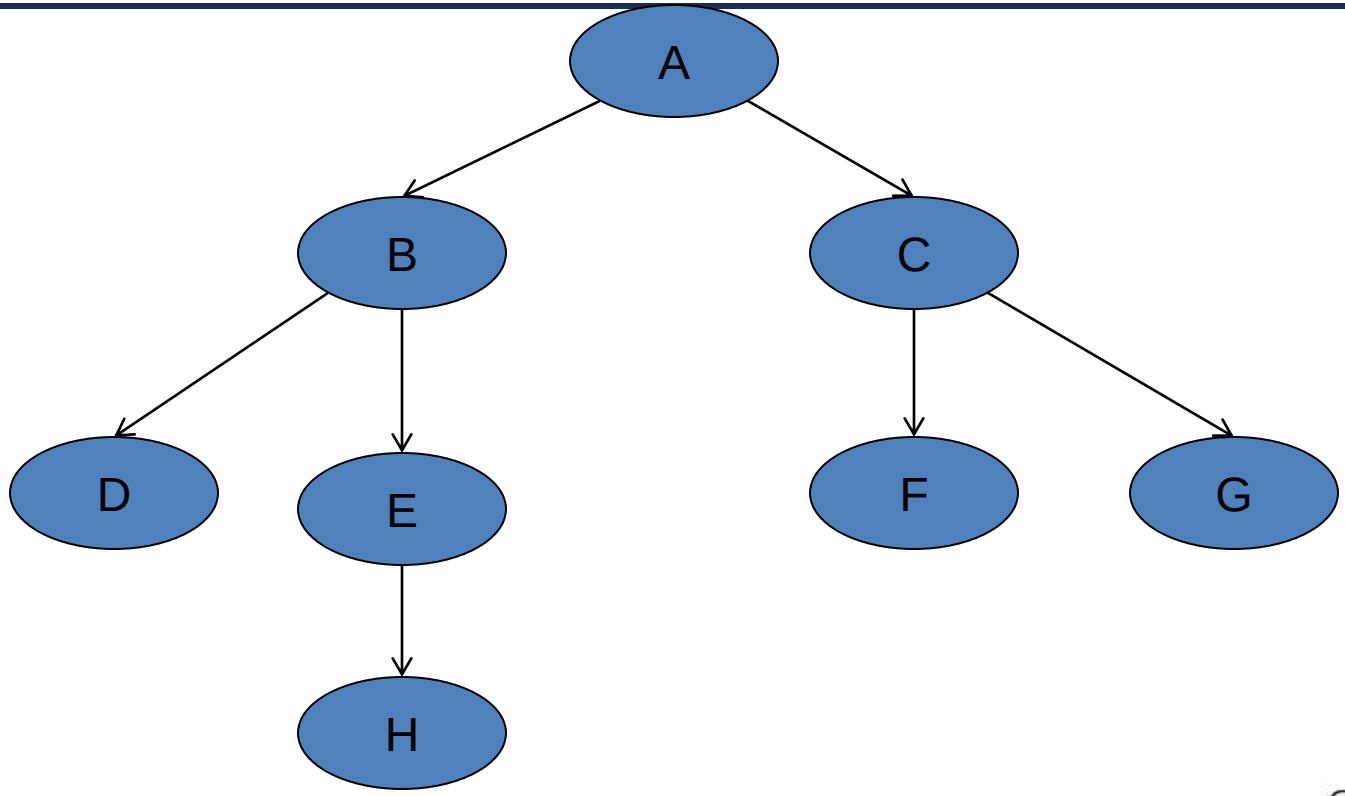
- Breadth - First Search
  - Used commonly for search engines with broad subjects (general purpose)
- Depth - First Search
  - Used commonly for subject specific search engines

How does a crawler visit web pages?

Breadth - First Search

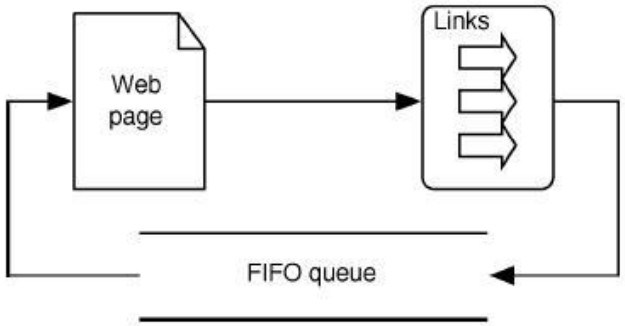
- In this traversal strategy, a node (page) and all of its “siblings” are visited first, before exploring any “children.”
- This strategy tends to discover high-quality pages early in the crawl, with the justification that most important pages have many links to them from numerous hosts, and those links will be found early, regardless of on which host or page the crawl originate.
- Similarly, breadth-first crawling leads to a more diverse dataset earlier on, rather than a depth-first approach which may end up traversing deep paths within a given site.

Breadth- First Search



Order of URL visits:

A→ B→ C→ D→ E→ F→G → H

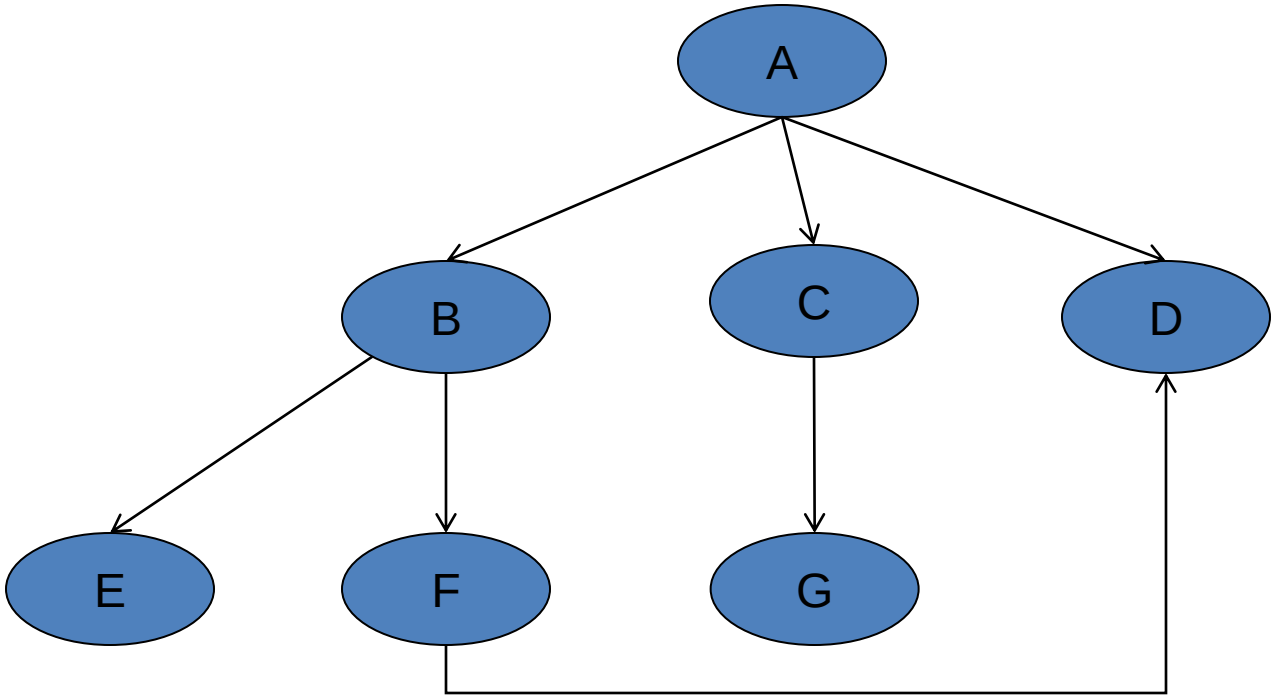


How does a crawler visit web pages?

Depth - First Search

- In this traversal strategy: start at a node (web page), pick one of the outgoing links from it, explore that link (and all of that link's outgoing links, and so on) before returning to explore the next link out of our original node.
- The name "depth-first" comes from the fact that we go down the graph before we go across.

Depth- First Search



Order of URL visits:

A→ B→ E→ F→ D→ C→G

World Wide Web Characteristics

- There are ***three*** important characteristics of the Web that make crawling it very difficult:
  - its large ***volume***
  - its fast rate of ***change***
  - ***dynamic*** page generation
- This produces a big variety of possible crawlable URLs.

Crawling problems

As a result of the WWW characteristics:

- A crawler downloads a fraction of the Web pages within a given time
  → ***prioritize downloads***

- While a crawler downloads the last pages from a site, it is very likely that new pages have been added to the site, or that pages have already been updated or even deleted
  → ***revisit pages***

Crawling policies

The behavior of a Web crawler is the outcome of a combination of ***policies***:

- *selection policy* that states which pages to download
- *re-visit policy* that states when to check for changes to the pages
- *politeness policy* that states how to avoid overloading Web sites
- *parallelization policy* that states how to coordinate distributed Web crawlers.

What any crawler *must* do

- Be Polite: Respect implicit and explicit politeness considerations
  - Only crawl allowed pages
  - Respect *robots.txt*
- Be Robust: Be immune to spider traps and other malicious behavior from web servers
  - A spider trap (or crawler trap) is a set of web pages that may intentionally or unintentionally be used to cause a web crawler or search bot to make an infinite number of requests (e.g., Dynamic calendar on a website).

Explicit and implicit politeness

- Explicit politeness: specifications from webmasters on what portions of site can be crawled
  - robots.txt
- Implicit politeness: even with no specification, avoid hitting any site too often
  - insert time gap between successive requests to a host that is >> time for most recent fetch from that host

What any crawler ***should*** do

- Be capable of distributed operation: designed to run on multiple distributed machines (*to reduce waiting time for responses to requests*).
- Be scalable: designed to increase the crawl rate by adding more machines.
- Performance/efficiency: permit full use of available processing and network resources.

What any crawler *should* do

- Fetch pages of "higher quality" first
- Continuous operation: Continue fetching fresh copies of a previously fetched page
- Extensible: Adapt to new data formats, protocols

Focused Crawling

- Attempts to download only those pages that are about a particular topic
  - used by *vertical search* applications
- Rely on the fact that pages about a topic tend to have links to other pages on the same topic
  - popular pages for a topic are typically used as seeds
- Crawler uses *text classifier* to decide whether a page is on topic

Sitemaps

- Sitemaps contain lists of URLs and data about those URLs, such as modification time and modification frequency
- Generated by web server administrators
- Tells crawler about pages it might not otherwise find
- Gives crawler a hint about when to check a page for changes

Sitemap Example

```
<?xml version="1.0" encoding="UTF-8"?>
<urlset xmlns="http://www.sitemaps.org/schemas/sitemap/0.9">
  <url>
    <loc>http://www.company.com/</loc>
    <lastmod>2008-01-15</lastmod>
    <changefreq>monthly</changefreq>
    <priority>0.7</priority>
  </url>
  <url>
    <loc>http://www.company.com/items?item=truck</loc>
    <changefreq>weekly</changefreq>
  </url>
  <url>
    <loc>http://www.company.com/items?item=bicycle</loc>
    <changefreq>daily</changefreq>
  </url>
</urlset>
```

Storing the Downloaded Documents

- Document data store
  - Stores text, metadata, and other related content for documents
    - Metadata is information about document such as type and creation date
  - Provides fast access to document contents for search engine components
    - e.g. result list generation
  - Could use relational database system
    - More typically, a simpler, more efficient storage system is used due to huge numbers of documents


WEB CRAWLERS
nutch
STORMCRAWLER
semrush
DEEPCRAWL


THANK YOU!